Deep Learning (BEV033DLE)
Lecture 10
Regularizers

Alexander Shekhovtsov

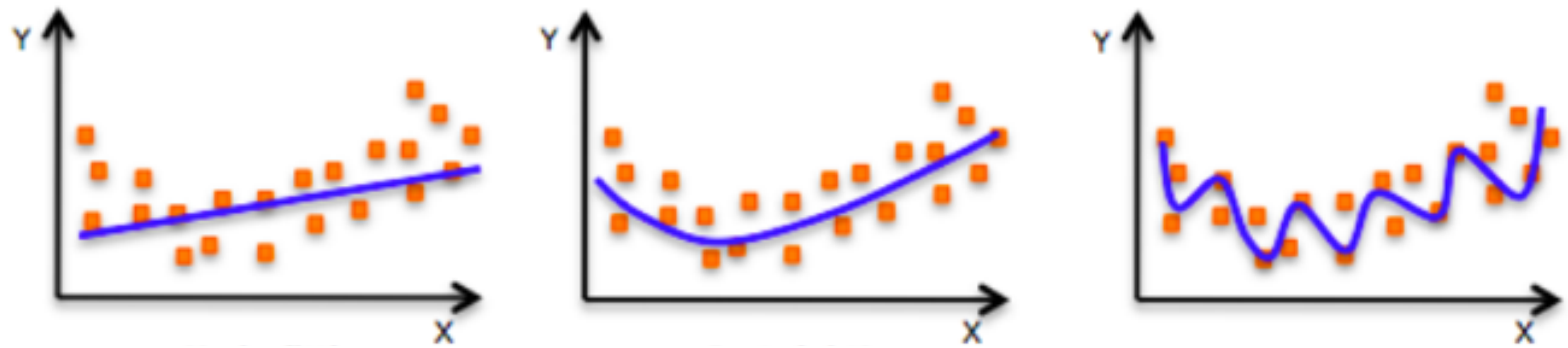
Czech Technical University in Prague

- ✦ L2 regularization (Weight Decay)
- ✦ Dropout
- ✦ Slightly Beyond
  - Other Norms
  - Batch Normalization
  - Implicit Regularization of SGD / SMD

Introduction (Overfitting)

Underfitting and Overfitting

✦ Classical view in ML:

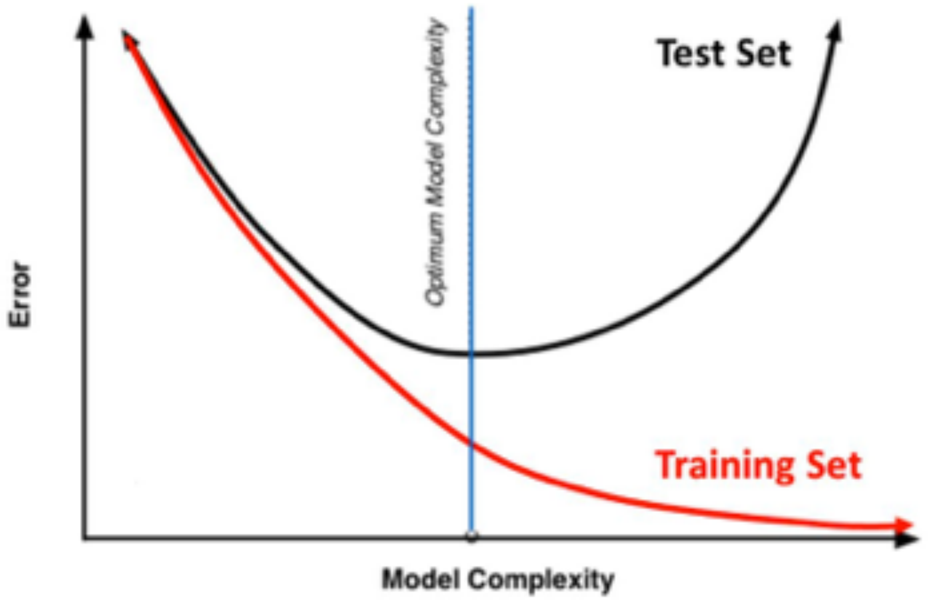


underfitting — capacity to low

just right

overfitting — capacity to high

Training Vs. Test Set Error

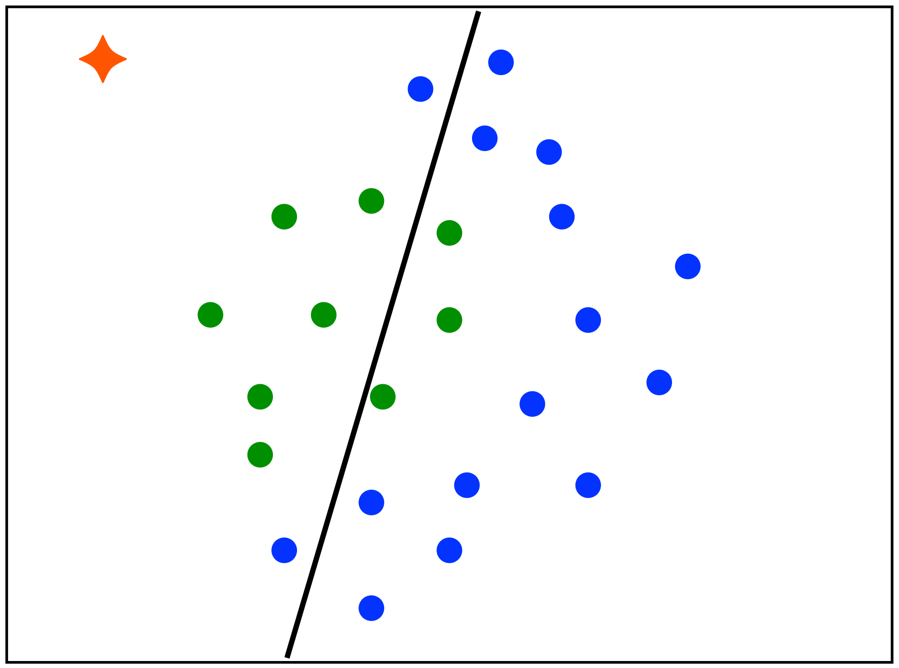


✦ Control model capacity (prefer simpler models, regularize) to prevent overfitting

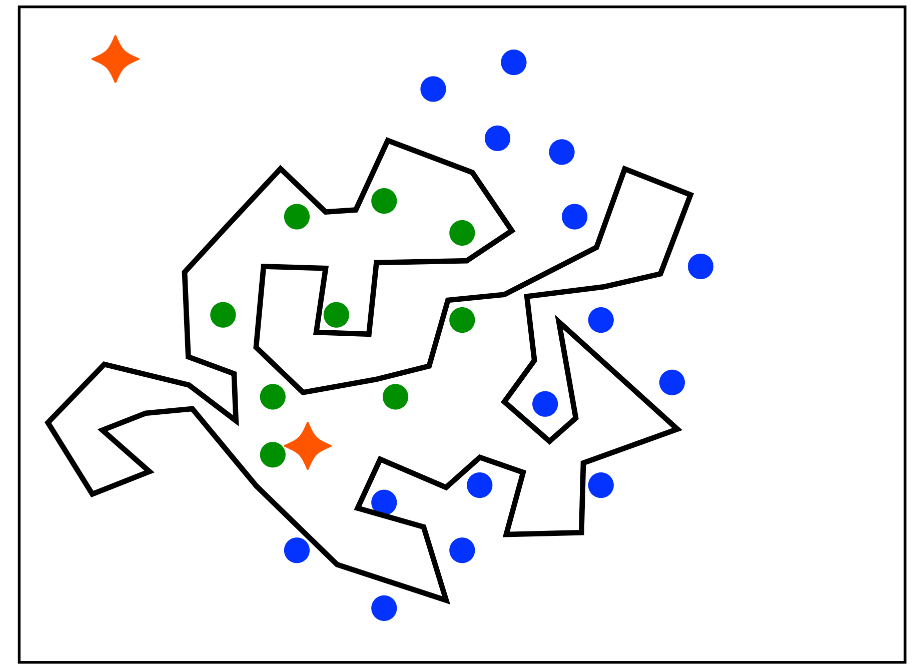
- in this example: limit the number of parameters to avoid fitting the noise

Underfitting and Overfitting

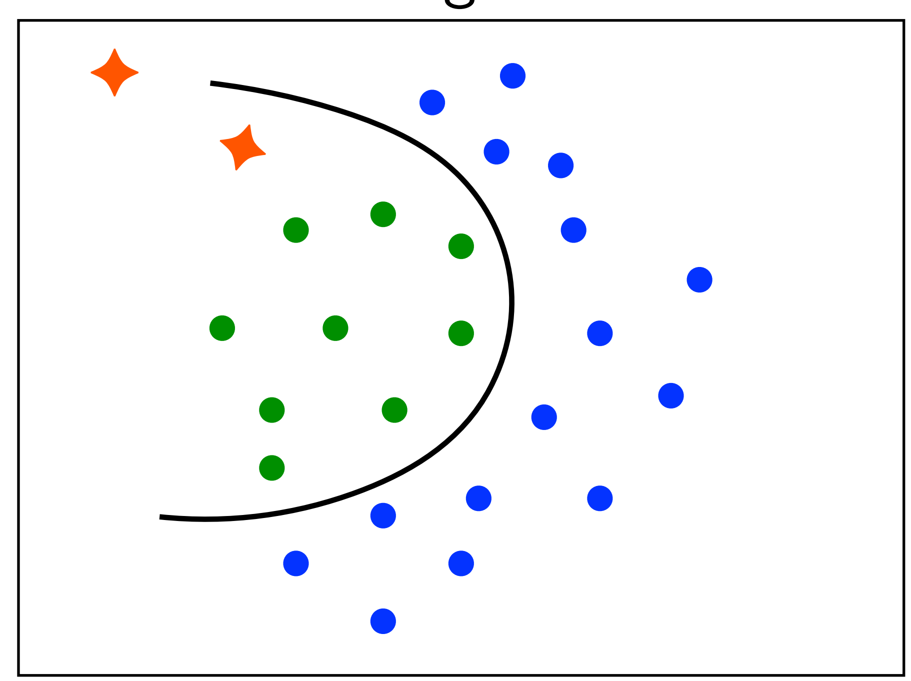
Underfitting — model capacity too low



Overfitting — model capacity too high



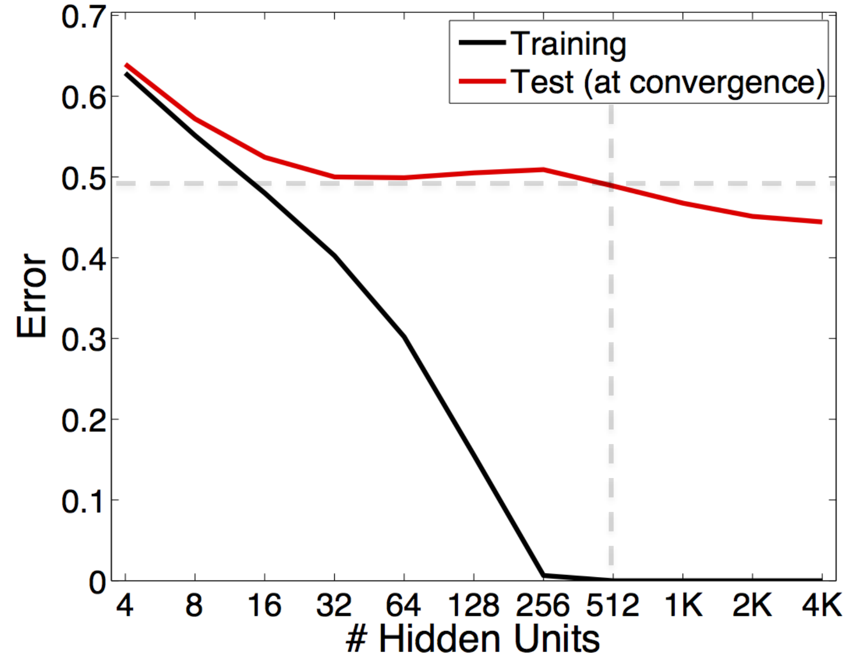
Good overfitting?



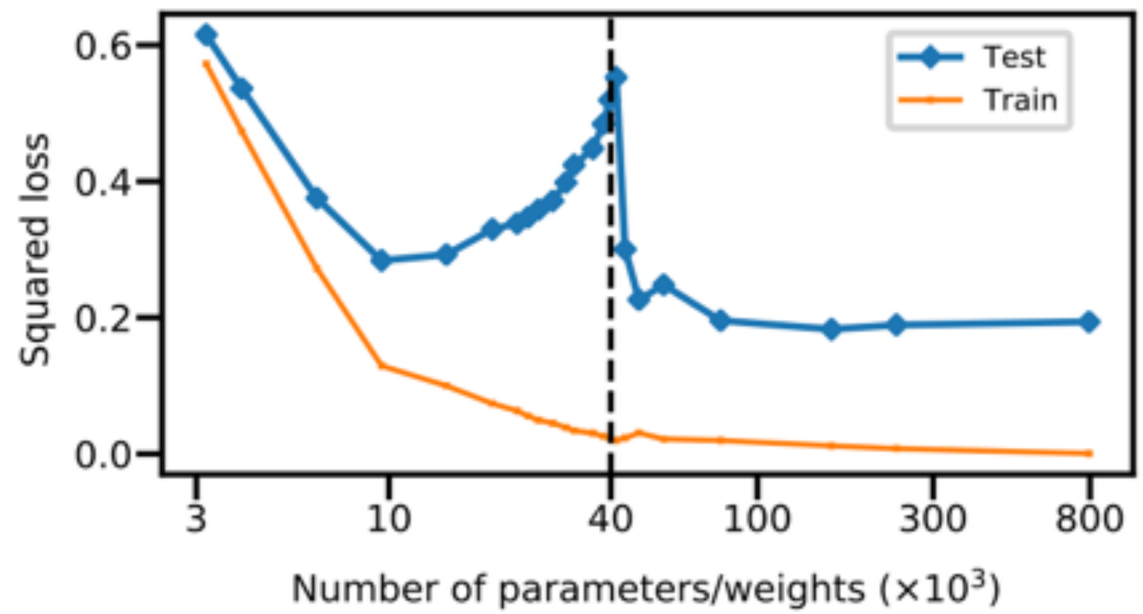
- Models in practice are chosen to perfectly fit training data (overparametrized)
- The boundary may be arbitrary complex as they can fit any labeling

Generalization of Over-Parametrized Models

- Right models + SGD generalize **better** in overparametrized regime



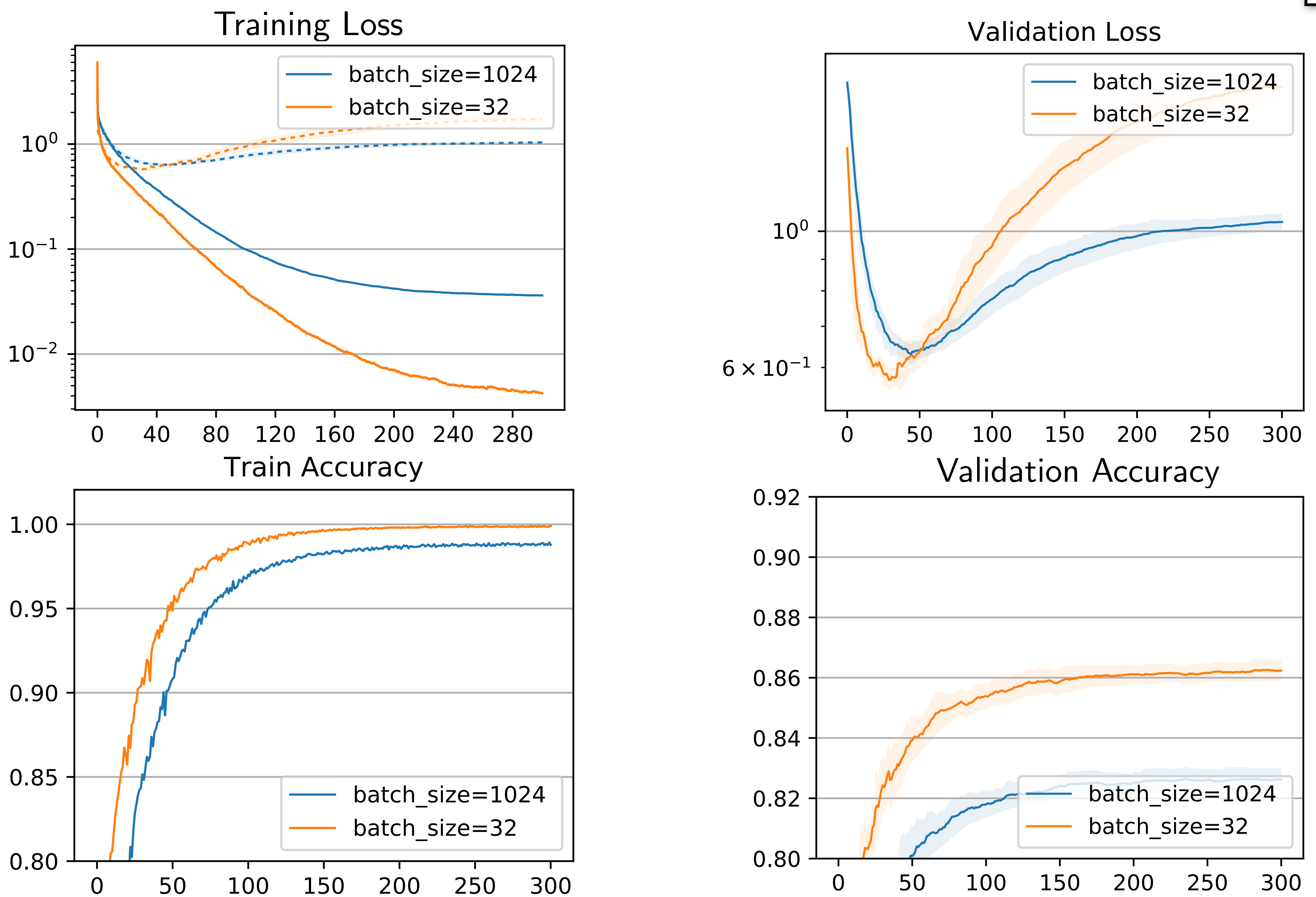
[Neyshabur (2015)]



[Belkin et al. 2019]

- Clearly regularizing by controlling the number of parameters is not the best option
- Important to regularize by other means:
  1. Good model architecture (putting our knowledge of invariances and useful information processing blocks into the network structure)
  2. Everything else counts as implicit regularization matters (optimizer, batch size etc.)
  3. Explicit regularization

Symptoms of Overfitting in Classification



- Training loss approaches 0
- Train accuracy goes to 100%
- Validation loss starts growing
- Validation accuracy still improves but calibration degrades

L_2 Regularization (Weight Decay)

General Setup

- Regularized training objective:

$$\min_\theta L(\theta) + \lambda R(\theta) = \min_\theta \sum_i l_i(y_i|x_i;\theta) + \lambda R(\theta)$$

  - $R(\theta)$ - function not depending on data
  - λ - regularization strength

- Recall connection to maximum a posteriori parameter estimation (MAP):

$$\max_\theta p(D|\theta)p(\theta)$$

  - $p(\theta) \propto \exp(-\lambda R(\theta))$ - prior on the model weights
  - $p(D|\theta)$ - likelihood of the data given parameters
  - $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$ - Bayesian posterior over parameters

RPZ lecture 3:(Parameter Estimation: Maximum a Posteriori (MAP))

- In practice, more commonly used as:

$$\min_\theta \frac{1}{n} \sum_i l_i(y_i|x_i;\theta) + \lambda R(\theta)$$

  - λ is tuned for a given dataset with cross-validation

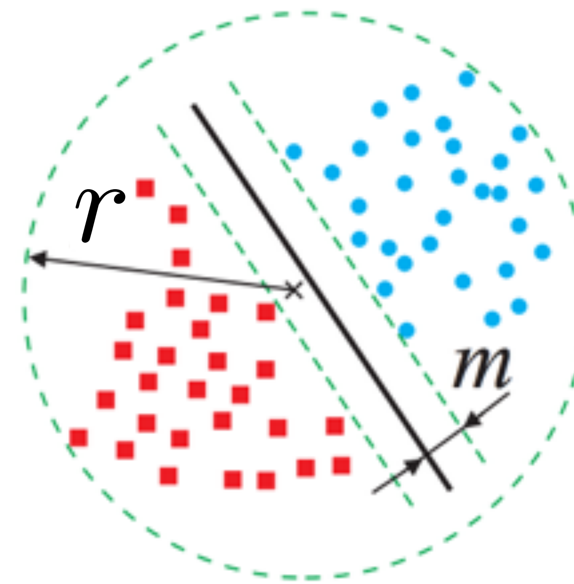
Background: Linear Models

- L_2-regularization (l_2, weight decay):

  $R(\theta) = \|\theta\|^2$

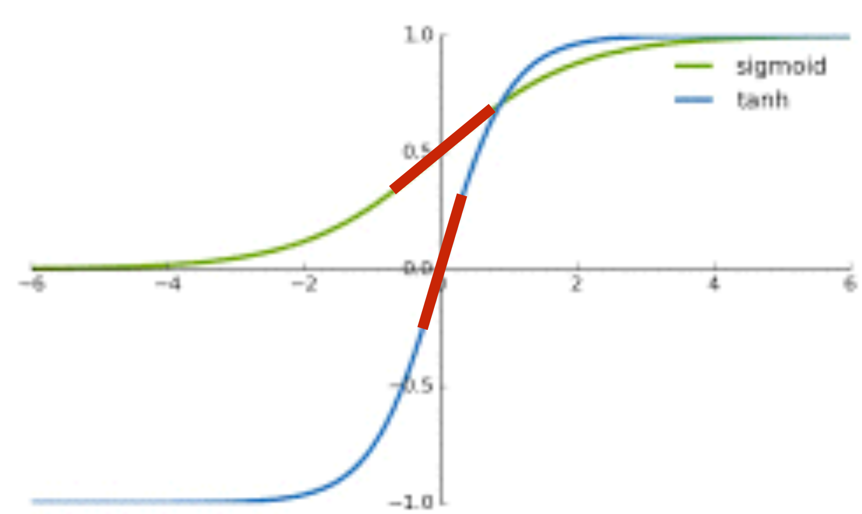
- In **linear regression**:
  - Known as ridge regression, Tikhonov regularization
  - Equivalent to using *multiplicative noise* $\mathcal{N}(1, \lambda^2)$ on the input
  - Smoothing effect (reduces the variance of $\hat{\theta}$)

- In **linear classification**:
  - Small $\theta \leftrightarrow$ large margin
  - Generalization bounds independent of dimensionality of the model (roughly): $\mathrm{Risk}(h) \leq O^*\left(\frac{1}{N}\frac{r^2 + \|\xi\|^2}{m^2}\right)$, where ξ are slacks

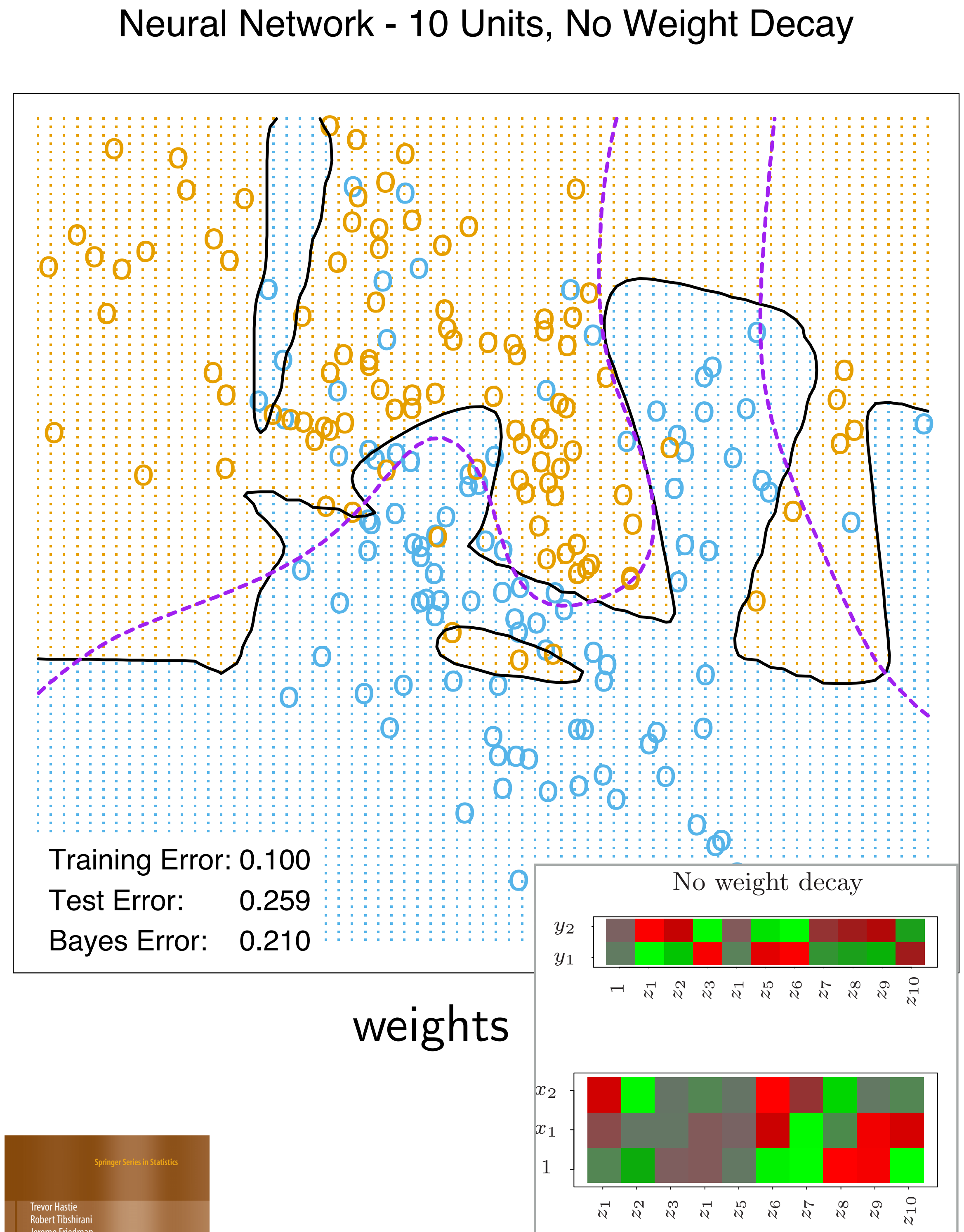


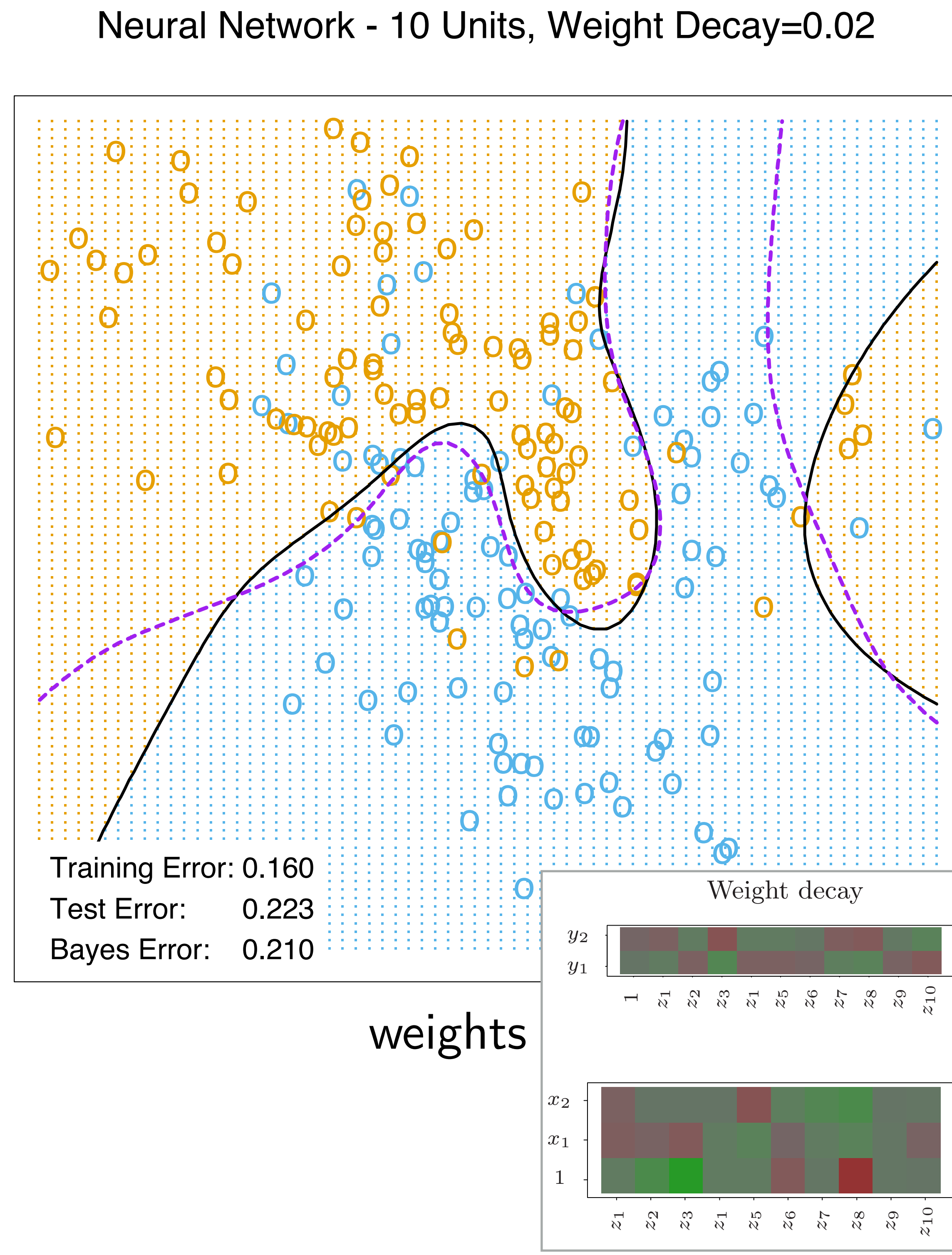
- **Sigmoid NNs**:
  - Small $\theta \rightarrow$ small activations

    $\rightarrow$ sigmoid outputs are close to linear



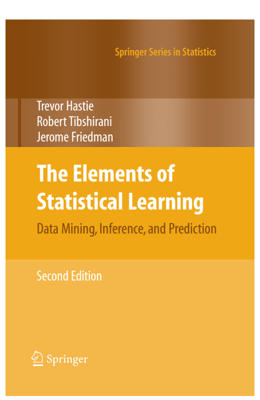
Example





weights

weights

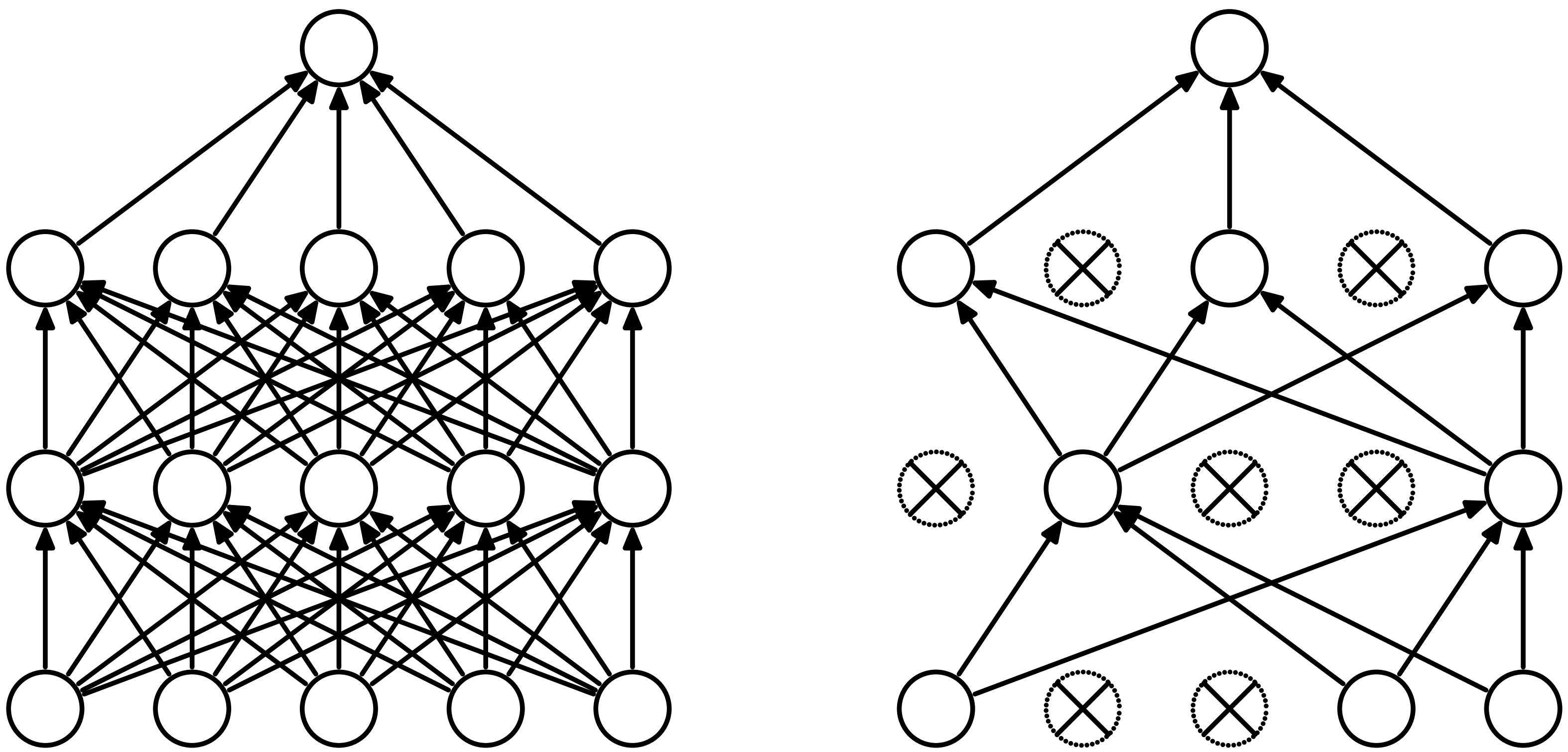


Hastie, Tibshirani and Friedman: The Elements of Statistical Learning

https://web.stanford.edu/~hastie/ElemStatLearn/

Dropout

Simple Idea

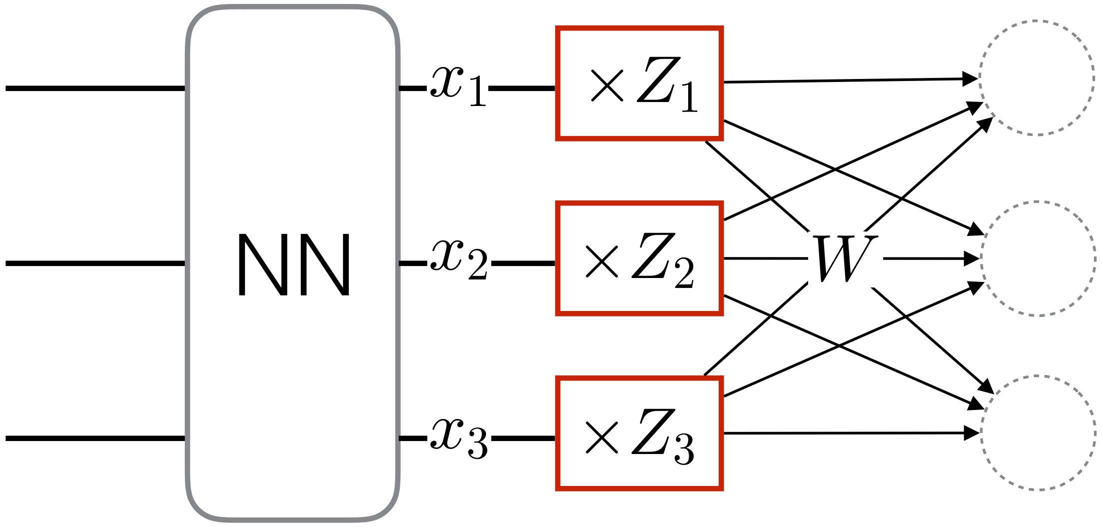

(a) Standard Neural Net (b) After applying dropout.

[Srivastava et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting]

- During training:
  - Randomly, make some units inactive by setting their outputs to zero
  - This results in the associated weights not being used and we obtain a (random) subnetwork
  - The network develops robustness to units being dropped
- During testing:
  - Use all units

Mathematical Model

- How we can model this:
  - Introduce random Bernoulli variables $Z_i = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1-p, \end{cases}$ multiplying outputs of the preceding layer
  - Can interpret outputs multiplied with 0 as dropped
  - Drop probability $q = 1 - p$
  - Next layer activations: $a = W(x \odot Z)$



$Z_i \sim$ Bernoulli(0.3)

- Prediction is random now?
  - Denote the network output as $f(x, Z; \theta)$
  - We have two choices how to make predictions:
    - **Randomized predictor:** $p(y|x, Z) = f(x, Z; \theta)$
    - **Ensemble:** $p(y|x) = \mathbb{E}_Z[f(x, Z; \theta)] = \sum_Z p(z) f(x, Z; \theta)$
- We randomized predictor for training (easier and other reasons)
- We will use ensemble (or its approximation) for testing

Note: Gaussian multiplicative $\mathcal{N}(1, \sigma^2)$ noises work as well (Gaussian Dropout)

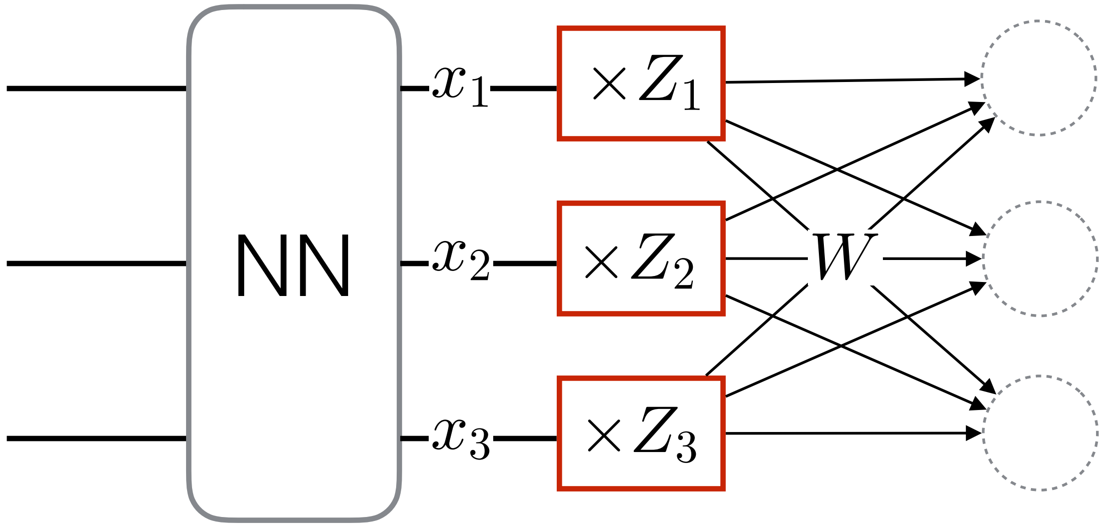
Training

- Loss of randomized predictor:
  - Double expectation in noises and date: $\mathbb{E}_Z\Big[\mathbb{E}_{(x,y)\sim\text{data}}\Big[l(y, f(x, Z; \theta))\Big]\Big]$
  - Same as: $\mathbb{E}_{Z\sim\text{Bernoulli}(q),\ (x,y)\sim\text{data}}\Big[l(y, f(x, Z; \theta))\Big]$
  - Unbiased loss estimate using a batch of size M:
    $\frac{1}{M}\sum_{i=1}^{M} l(y_i, f(x_i, z_i; \theta))$
- What it means practically:
  - Draw a batch of data
  - For each data point i independently sample noises z
  - Compute forward and backward pass as usual
  - Will have increased variance of the stochastic gradient

Testing

- Use approximation (common default):
  - $\mathbb{E}_Z\Big[f(x, Z; \theta)\Big] \approx f(x, \mathbb{E}_Z[Z]; \theta)$
  - Since $\mathbb{E}_Z[Z] = p$, we have

    $a = W(x \odot \mathbb{E}[Z]) = (pW)x$
  - i.e. need to scale down the weights



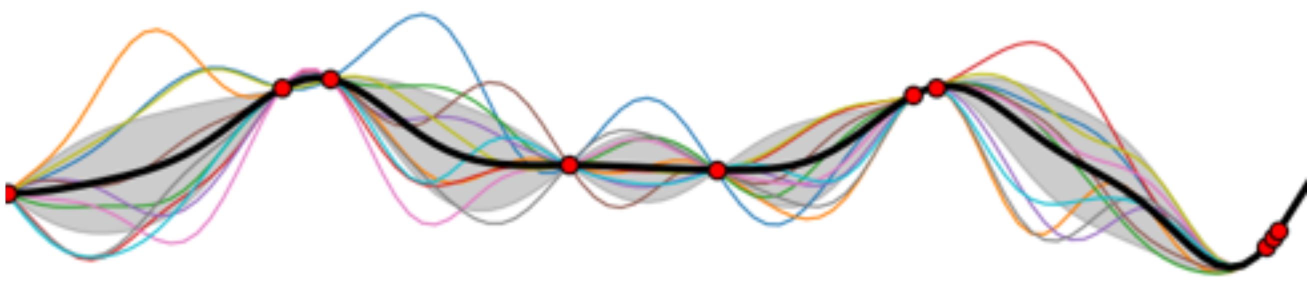
$Z_i \sim$ Bernoulli(0.3)

$E[Z] = p$

- Use sampling:
  - $\mathbb{E}_Z\Big[f(x, Z; \theta)\Big] \approx \frac{1}{M}\sum_{i=1}^{M} f(x_i, z_i; \theta)$
  - Generalizes slightly better than the above
  - Can be used to also estimate model uncertainty

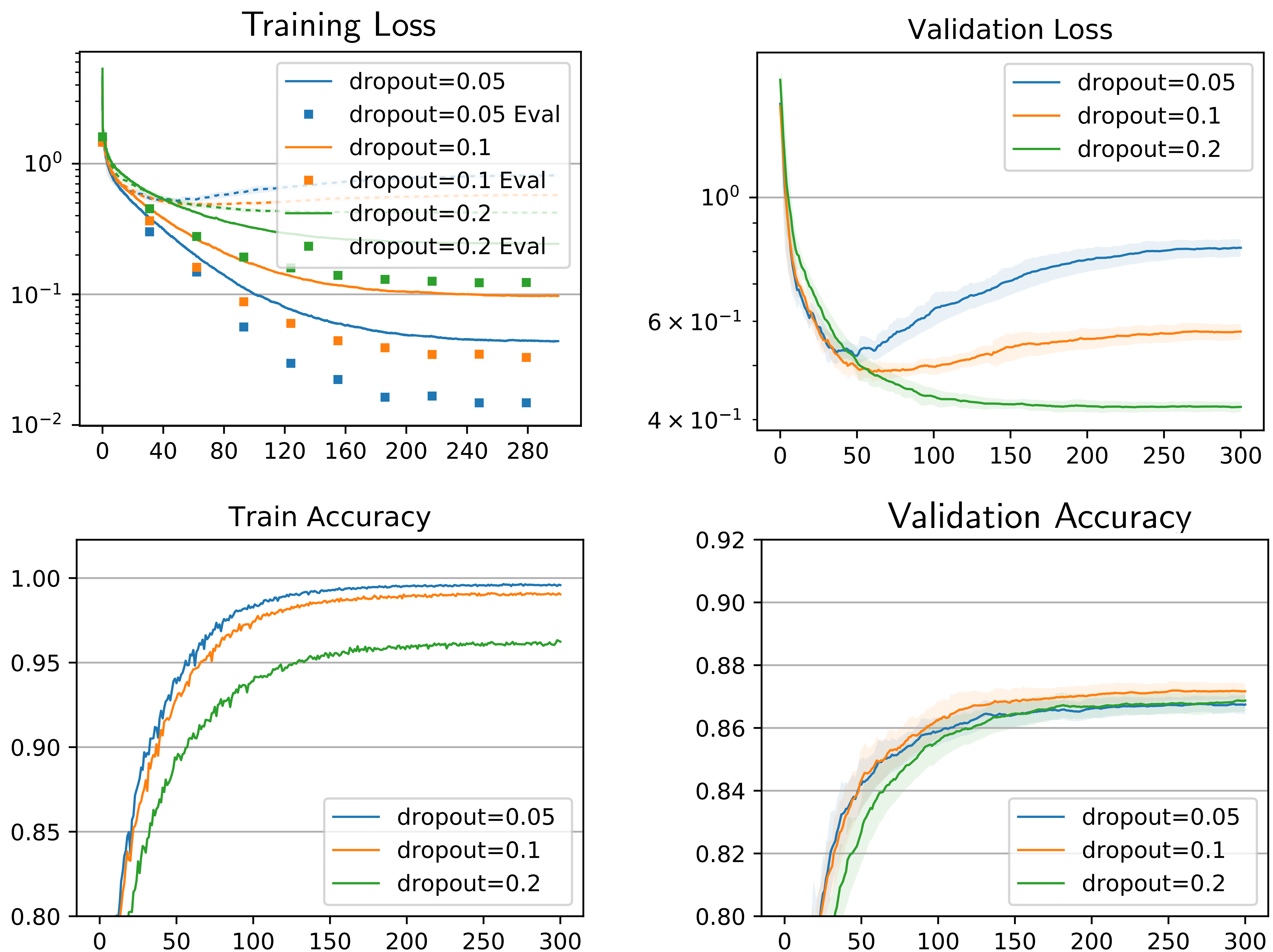
- Both variants achieve a "comity" or "ensembling" effect

averaging of many well fitting models:



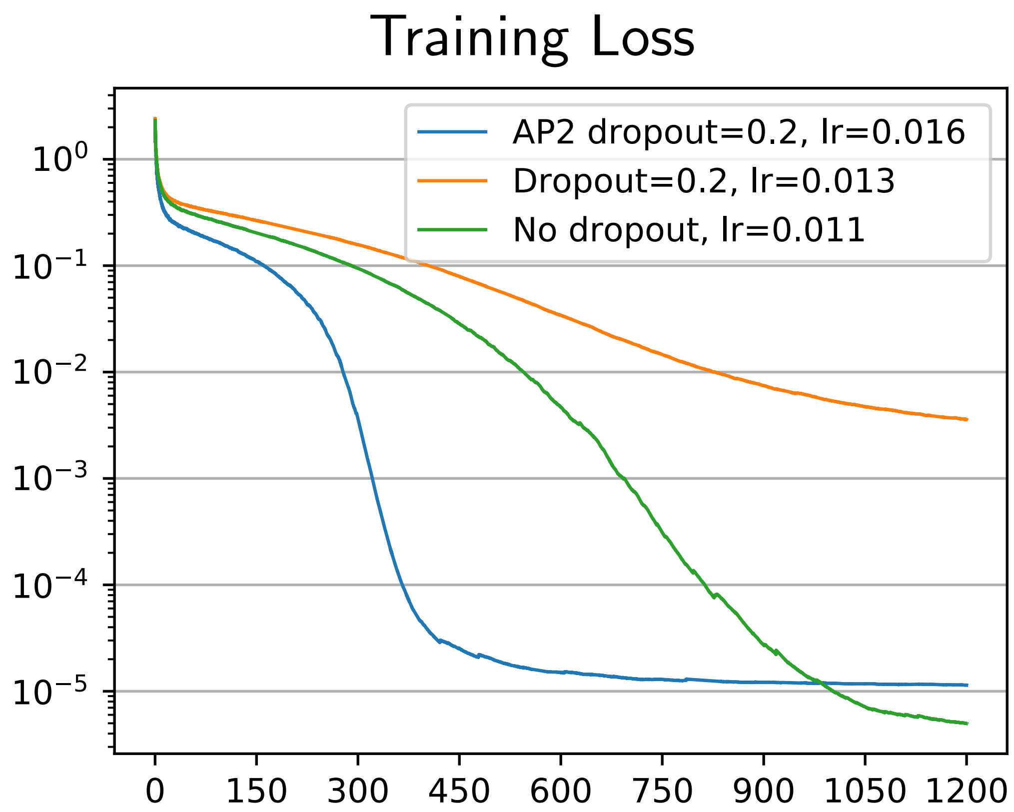
- More accurate analytic approximations than the first option are possible

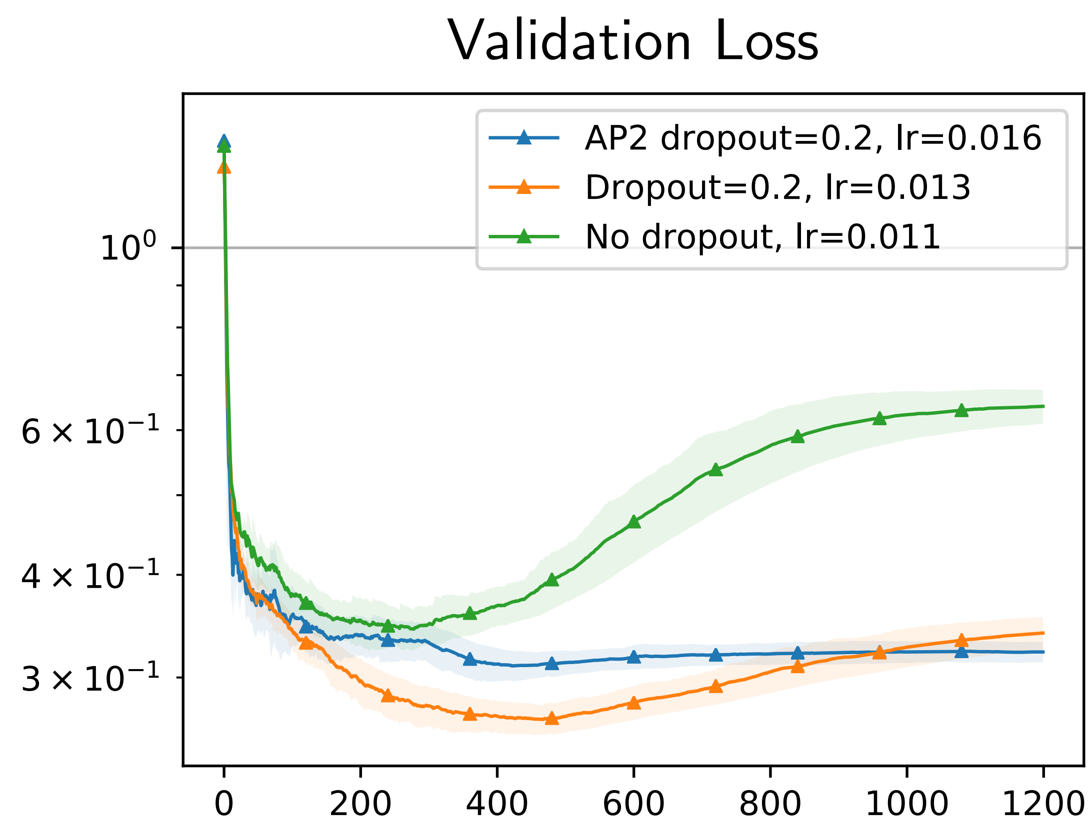
Example: Applying Dropout



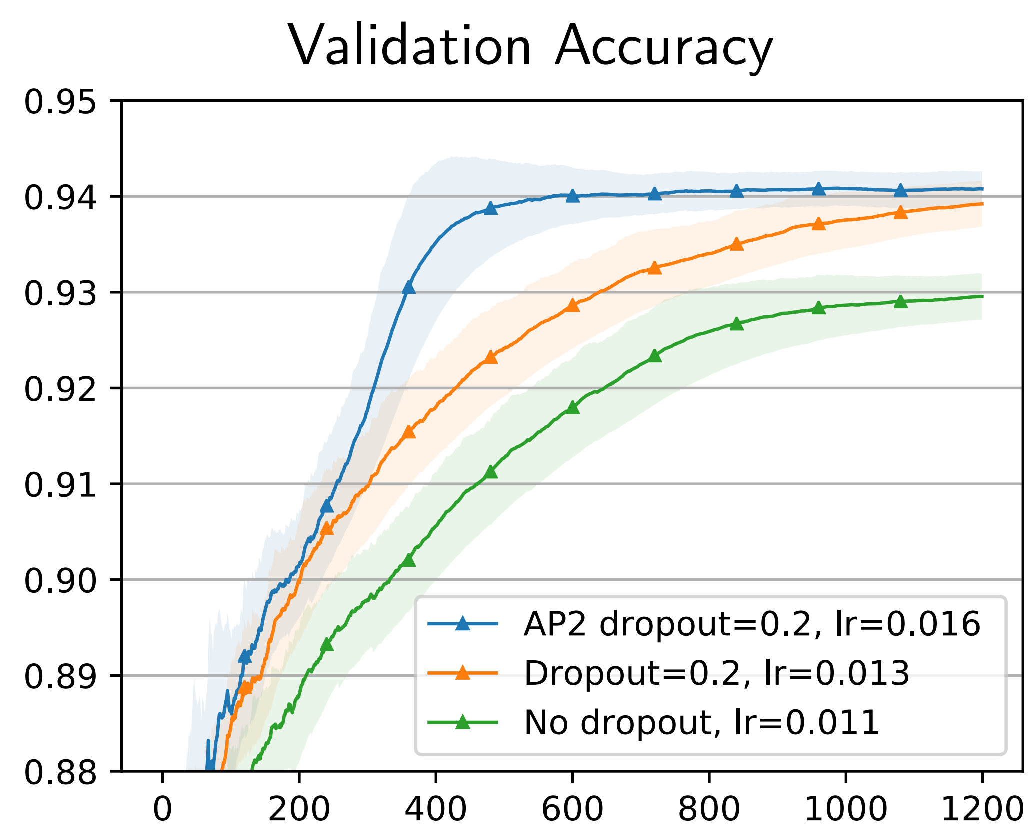
- Here it looks like it did not help with the validation accuracy, but see next slide

Example: Applying Dropout



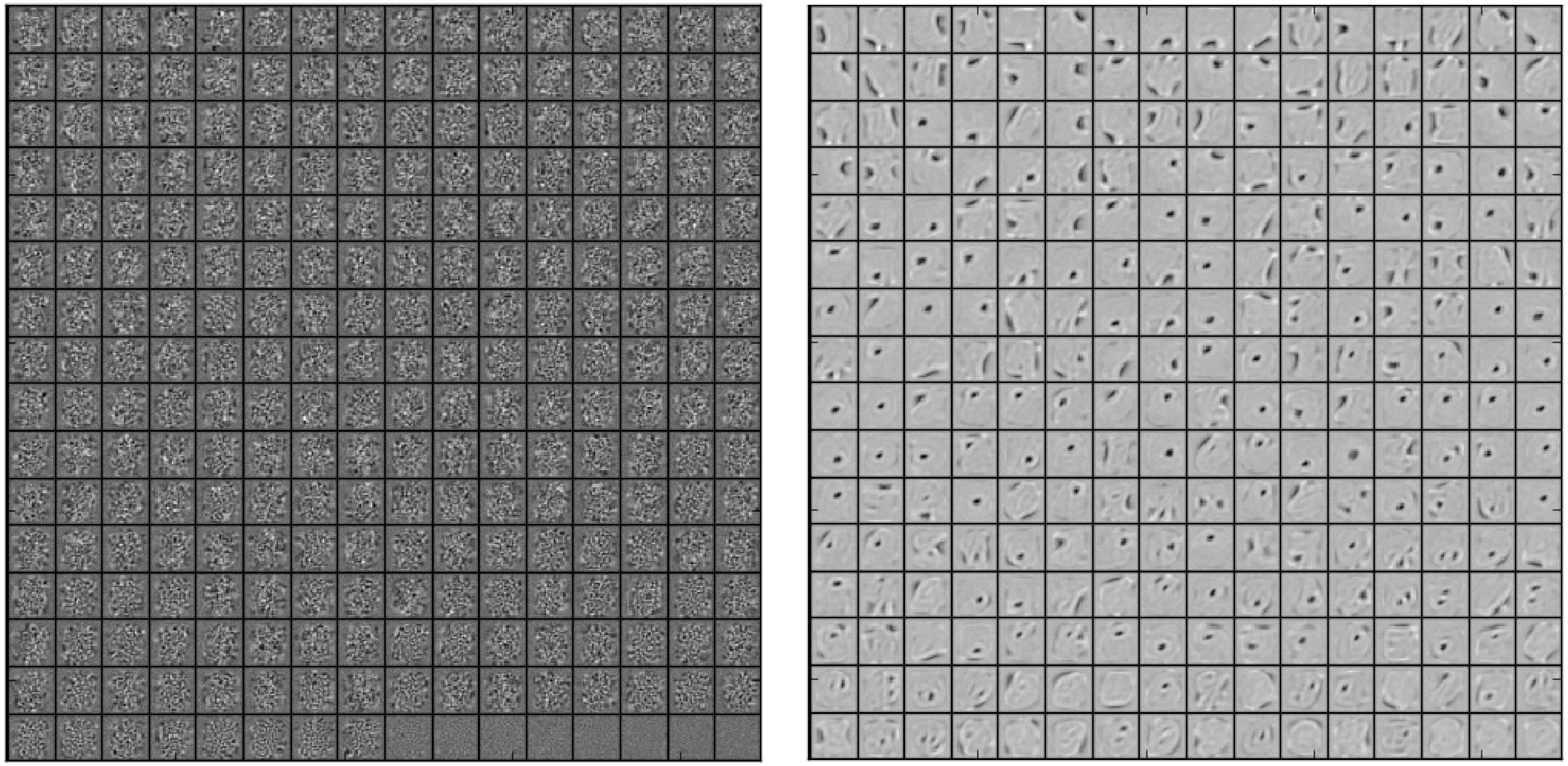


- Change the learning setup:
  - train longer with a slower learning rate decay
- Now it works!
  - There are (advanced) techniques to approximate it analytically: Fast Dropout, Analytic Dropout



Effect on Features

- Experiment:
  - MNIST auto encoder with 1 fully-connected hidden layer of 256 units

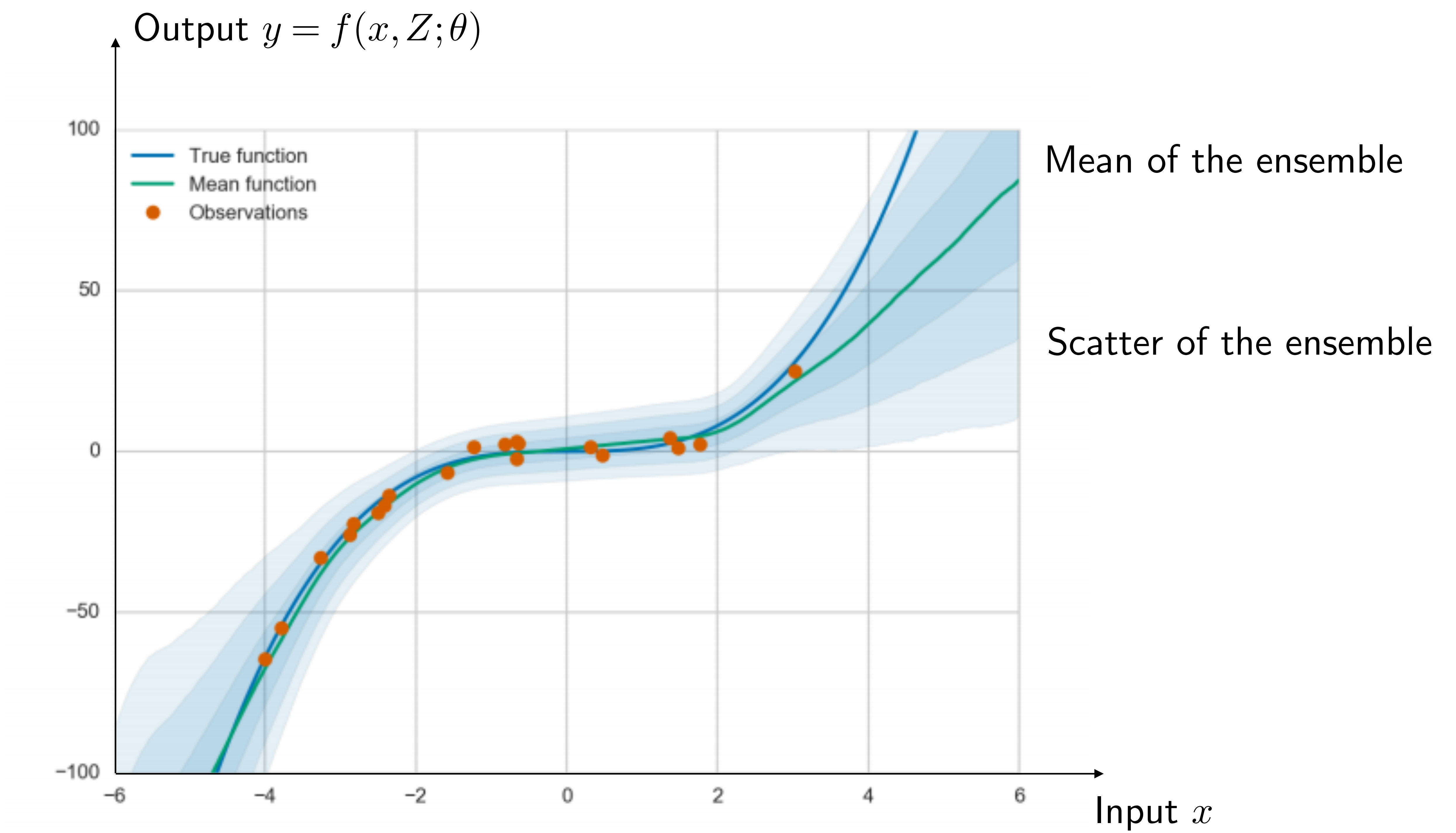


(a) Without dropout (b) Dropout with $p = 0.5$.

[Srivastava et al. (2014)]

- Hypothehis: dropout prevents co-adaptation of features and instead learns simpler features
- More interesting studies in the paper: effect on activation sparsity, connection to ridge regression, etc.

Model Uncertainty with Dropout



[Louizos and Welling 2017]

Beyond L_2 and Dropout

L_2 Regularization and Batch Normalization

- Consider BN-normalized layer:

  $a = \frac{Wx+b-\mu}{\sigma}\gamma + \beta$

  - $\mu = \frac{1}{M}\sum_i (Wx_i + b) \qquad \sigma^2 = \frac{1}{M}\sum_i (Wx_i + b - \mu)^2$
  - Exercise: the value of a does not depend depend on the bias b and the scale of the weights $W \to sW$

- What will happen if we try to solve $\min\limits_W L(a(W)) + \|W\|^2$, where $L(a(W))$ is invariant w.r.t. $\|W\|$?

L_2 Regularization and Batch Normalization

- Consider BN-normalized layer:

  $a = \frac{Wx+b-\mu}{\sigma}\gamma + \beta$

  - $\mu = \frac{1}{M}\sum_i (Wx_i + b)$ $\quad \sigma^2 = \frac{1}{M}\sum_i (Wx_i + b - \mu)^2$
  - Exercise: the value of a does not depend depend on the bias b and the scale of the weights $W \to sW$

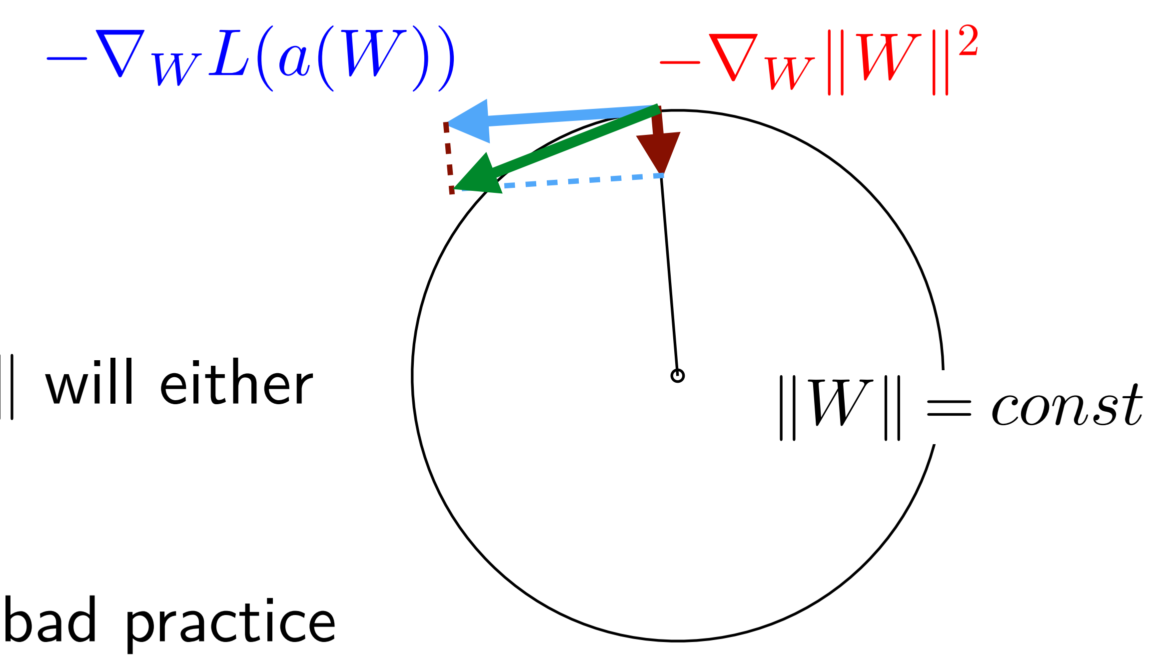
- What will happen if we try to solve $\min_W L(a(W)) + \|W\|^2$, where $L(a(W))$ is invariant w.r.t. $\|W\|$?
  - Make no sense, optimum value is approached with $\|W\| \to 0$

L_2 Regularization and Batch Normalization

- Consider BN-normalized layer:

  $a = \frac{Wx+b-\mu}{\sigma}\gamma + \beta$

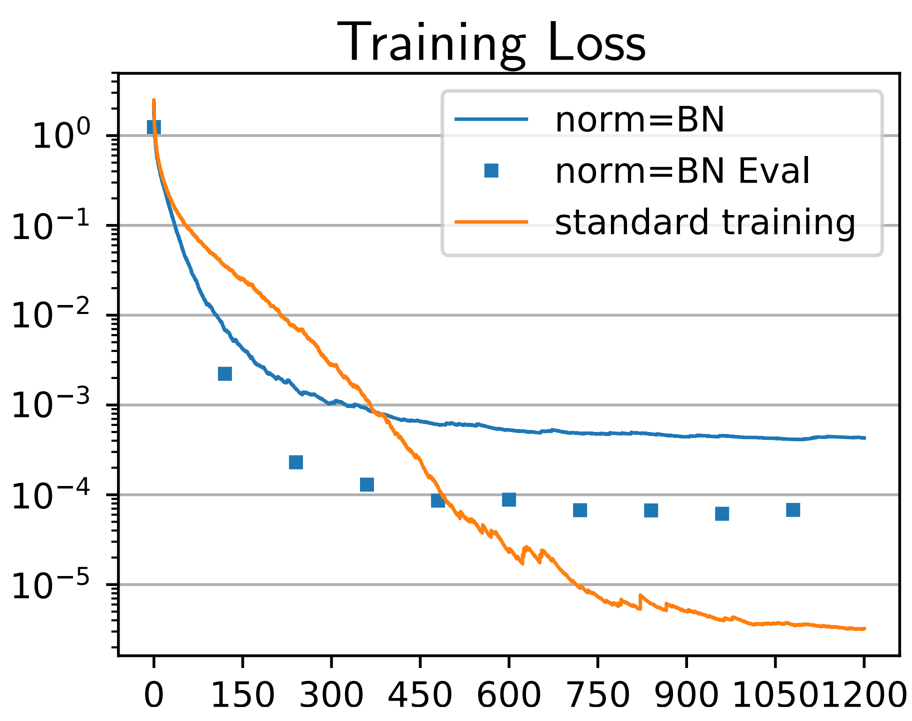
  - $\mu = \frac{1}{M}\sum_i (Wx_i + b) \qquad \sigma^2 = \frac{1}{M}\sum_i (Wx_i + b - \mu)^2$
  - Exercise: the value of a does not depend depend on the bias b and the scale of the weights $W \to sW$
- What will happen if we try to solve $\min_W L(a(W)) + \|W\|^2$, where $L(a(W))$ is invariant w.r.t. $\|W\|$?
  - Make no sense, optimum value is approached with $\|W\| \to 0$

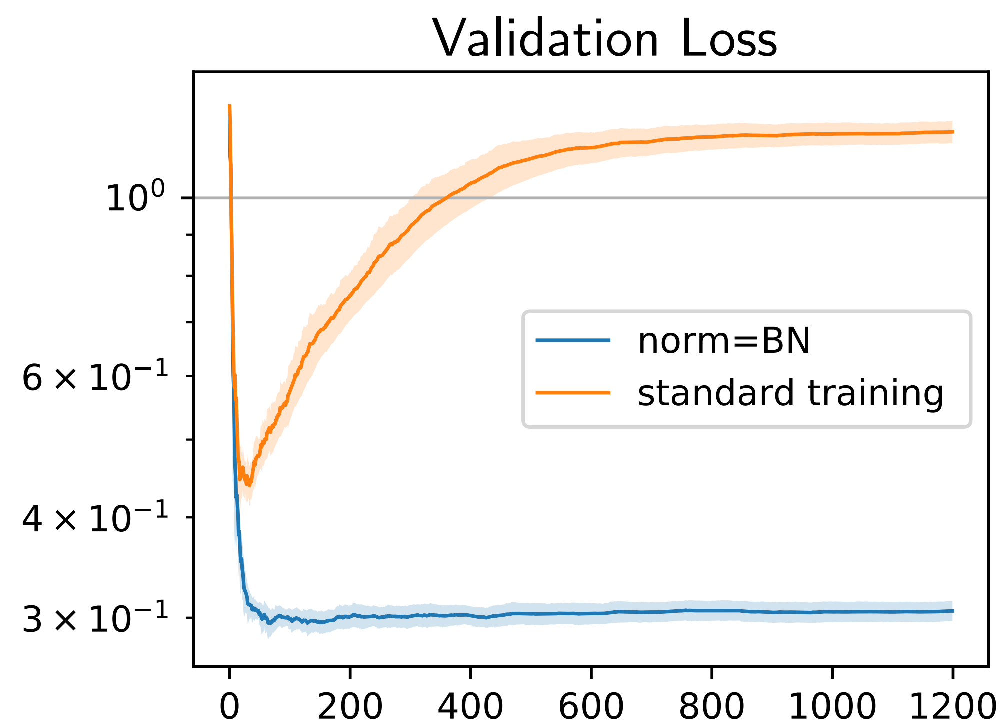


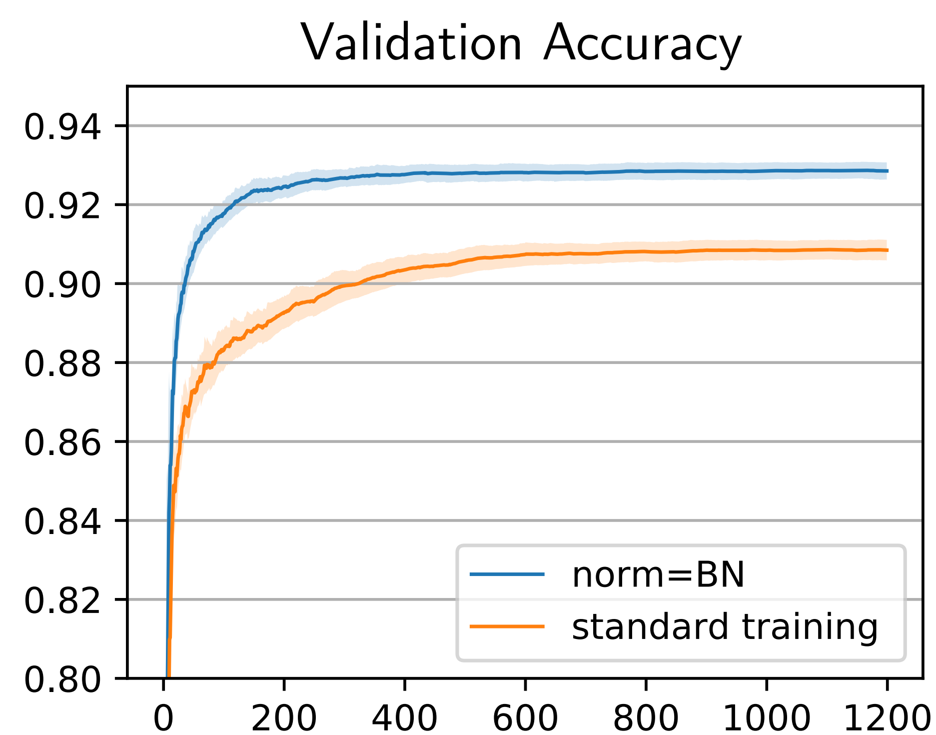
- GD iterates may still behave well
  - Actually, depending on λ, the norm $\|W\|$ will either grow or shrink during GD iterates
  - Possible to fiddle on this balance, but a bad practice

Batch Normalization Regularizes

- BN has rather strong regularization properties on its own (it depends on a randomly formed batch)



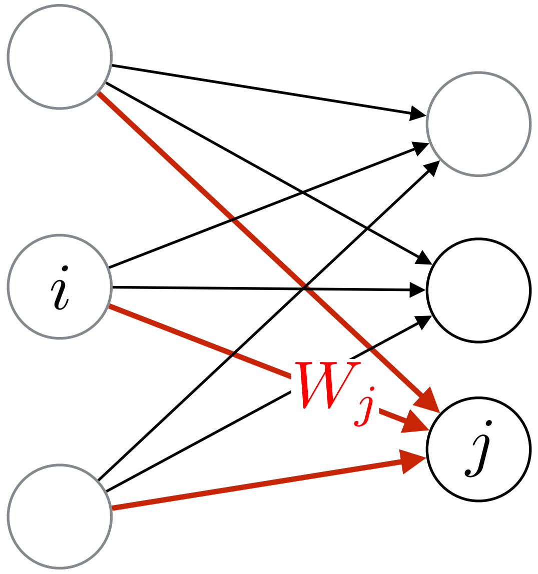




Other Norms

- $\mathbf{L}_1$ regularization: $R(W) = \|W\|_1 = \sum_{ij} |W_{ij}|$
  - Promotes sparsity
  - For better generalization we typically do not want sparsity (= less parameters)
- **Constrained** optimization form instead of penalty:

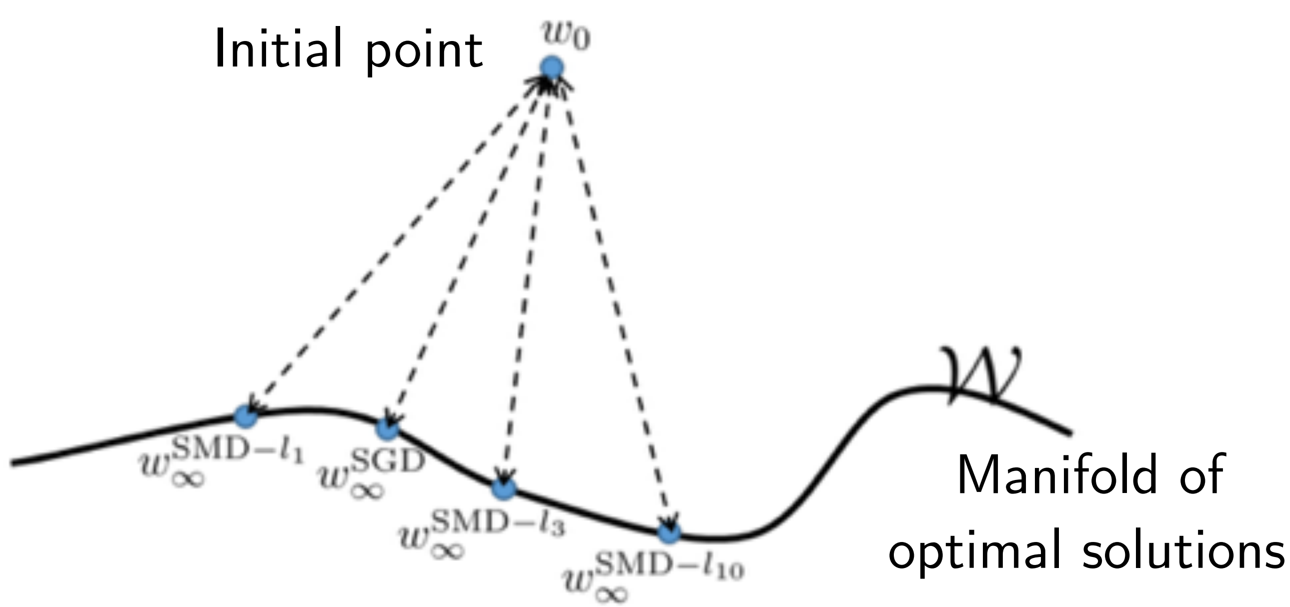
  $\min_W L(W)$ s.t. $R(W) \leq s$
  - Does not makes weights small, but prevents them from growing high
  - Can use projected SGD to solve
  - In particular L_2 norm on each column: $R(W) = \max_j \|W_j\|_2^2$ called **max-norm** appears useful

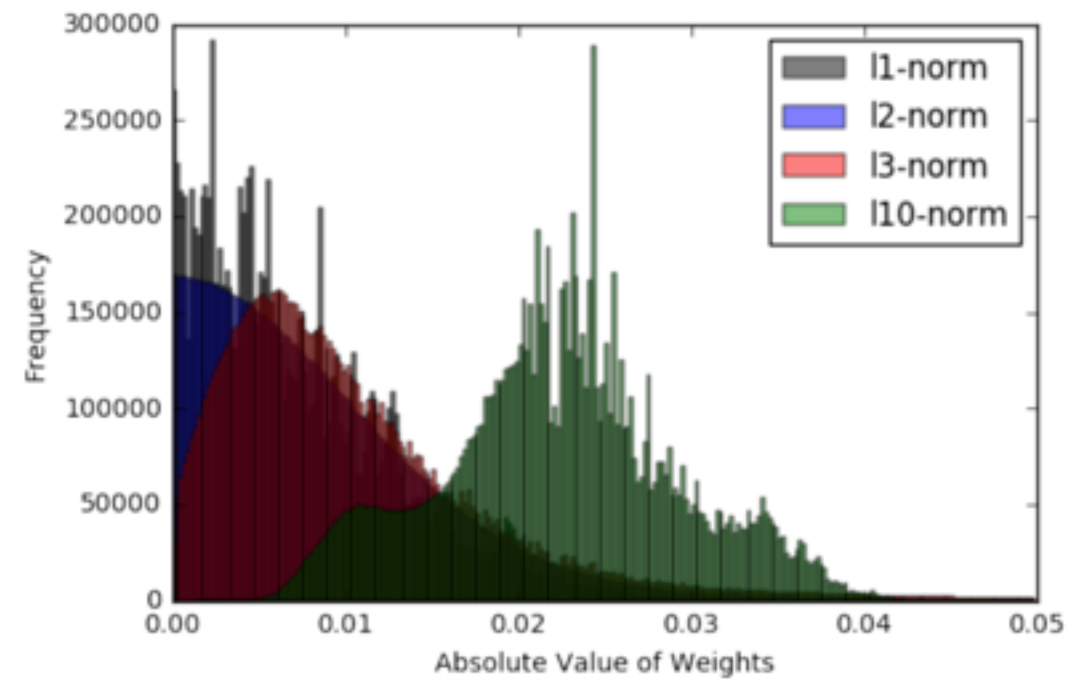


- **Generalizations**:
  - Flat L_p norm: $R(W) = \left(\sum_{ij} W_{ij}^p\right)^{\frac{1}{p}}$
  - **Group-norm**: $R(W) = \left(\sum_j \left(\sum_i W_{ij}^p\right)^{\frac{q}{p}}\right)^{\frac{1}{q}}$
  - Above variants are special cases
  - Different generalization bounds derived measuring complexity with group norm

Implicit Regularization by SGD / SMD

- Consider step proximal problem: $\min_x \langle \nabla f(x_0), x - x_0 \rangle + \lambda \|x - x_0\|_p^p$
  - i.e., p-norm stochastic mirror descent
- Using different p leads to solutions with different properties

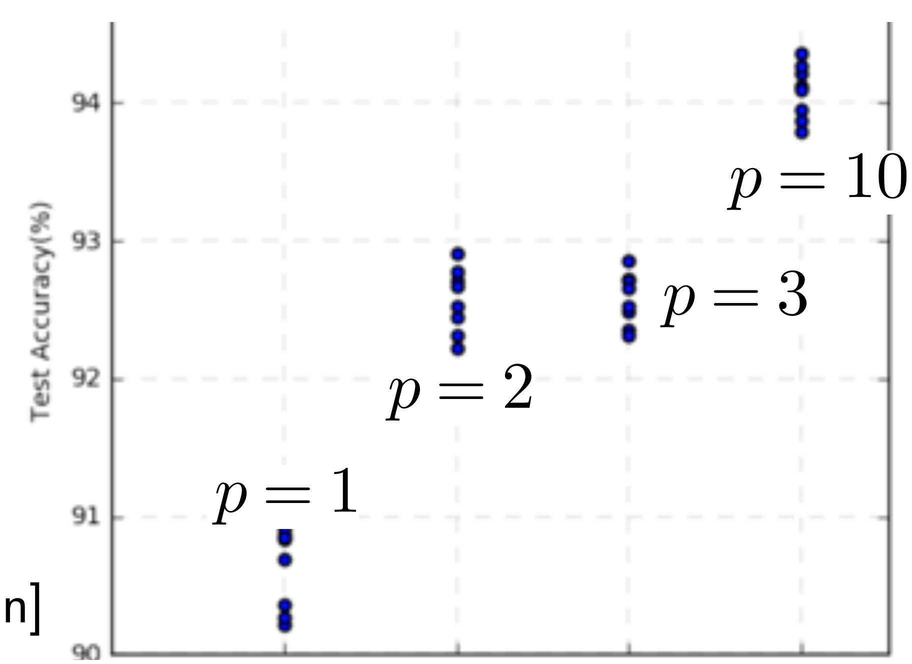




- Iterates tend to $\operatorname{argmin}_{w \in \mathcal{W}} \|w - w_0\|_p^p$, the closest point in the respective norm

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	141	9.19×10^3	4.1×10^4	2.34×10^5
2-norm BD	3.15×10^3	562	1.24×10^3	6.89×10^3
3-norm BD	4.31×10^4	107	53.5	1.85×10^2
10-norm BD	6.83×10^{13}	972	7.91×10^{-5}	2.72×10^{-8}

- Different sparsity and generalization



[Azizan et al. (2019) Stochastic Mirror Descent on Overparameterized Nonlinear Models: Convergence, Implicit Regularization, and Generalization]

Conclusion

- The most powerful regularization might be the network structure (inductive bias)
- In the overparametrized mode need to regularize
  - norms of the weights
  - data augmentation
  - activation augmentation / norm

- Some practical hints:
  - In convolutional layers BN is preferred to dropout. It also does something random that makes it generalize better and training is much faster
  - Do not combine BN with Weight Decay in same layers
  - Do not combine BN with Dropout in same layers

- We touched neural networks with noises
  - Deep topic: ensembles, Bayesian neural networks, expectation problems, stochastic and analytic approximations